Subject: Date:

«مبانی کامپیوتر»

نسل های کامپیوتر: ۱- نسل اول: اوایل دهه ۱۹۵۰ ← لامپ خلاء
۲- نسل دوم: اوایل دهه ۱۹۶۰ ← استفاده از ترانزیستور ها به جای لامپ خلاء
۳- نسل سوم: (در سال) ۱۹۶۴ ← مدارات مجتمع IC به جای ترانزیستور ها، کاهش حجم
۴- نسل چهارم: دهه ۱۹۷۰ ← مدارات فشرده IC ← حجم کم ترتیب به نسل قبل به فاندا رایانه خانه
۵- نسل پنجم ← کامپیوترهای هوشمند ۱۹۷۵ به بعد

نمای کلی یک سیستم کامپیوتری ⟸ خروجی و اطلاعات پردازش ⟵ داده ورودی
ورودی ↓ موس، کیبرد، ...
اطلاعات ↓ دادن نیت امیش
چون هوش پردازش صورت گرفته

اطلاعات: مجموعه مطالبی که وارد سیستم های کامپیوتری می‌شود و روش پردازش می‌شد و به شکل خروجی تبدیل می‌شد

هر سیستم کامپیوتری از سخت افزار و نرم افزار تشکیل می‌شود که جهت انجام کاری خاص با هم همکاری می‌کنند.

داده: مجموعه مطالبی که وارد کامپیوتر می‌شوند که می‌تواند به صورت صدا، حرف، یا تصویر یا فیلم باشد

پردازش: به کلیه فعالیت هایی که بر روی داده ها انجام می‌گیرد که منجر به پیدایش نتایج می‌گردد.

اطلاعات: به خروجی تولید شده بعد از پردازش داده اطلاعات گفته می‌شود

الگوریتم: دستورالعمل هایی برای انجام کاری توسط کامپیوتر نوشته می‌شوند را الگوریتم می‌گویند

برنامه کامپیوتری: به تشریح الگوریتم برای کامپیوتر با استفاده از یک زبان برنامه سازی، برنامه کامپیوتری گویند.

زبان برنامه سازی: زبانی که برای کامپیوتر قابل فهم باشد و الگوریتم ها با استفاده از آن به کامپیوتر داده می‌شود.

Subject: Date:

زبان کامپیوتر
- نسل اول ← ۰ و ۱ مستقیماً سمت
- ″ دوم ← زبان اسمبلی نمادها جایگزین شدند.
- ″ سوم زبان دست بالا
- ″ چهارم زبان سطح بالاتر
- ″ پنجم شی گرا کدی بهتر راحت تر

زبان نسل اول ← زبان ماشینی ← مستقیماً به زبان خود کامپیوتر نوشته می شود و توسط خود کامپیوتر قابل اجراست. در این زبان از ۰ و ۱ استفاده می شود

زبان نسل دوم زبان اسمبلی است که حالت نمادین زبان ماشین است در این دستورات با استفاده از یک نماد به جای صفر و یک نوشته می شود. زبان های ماشین و اسمبلی زبان های سطح پایین هستند.

زبان نسل سوم این نسل شامل زبان های سطح بالا مانند ++C و FORTRAN و Pascal و ... می باشد برنامه نویسی به این زبان ها بسیار نزدیک به زبان انسان است و از دستورات مشابه زبان طبیعی اغلب زبان انگلیسی تشکیل شد.

زبان نسل چهارم این نسل شامل زبان های سطح بالا است مانند زبان اسکیوئل یا SQL که زبانی خاص و برای منظوری خاص است این زبان برای دریافت اطلاعات از پایگاه داده استفاده می شود

زبان نسل پنجم این زبان ها شامل محیط های گرافیکی مناسب و راحت برای تولید نرم افزار ها هستند
Visual C#
sharp

انواع کامپیوتر ها
1. super computer ابر کامپیوترها
2. main frame کامپیوترهای بزرگ
3. mini computer کامپیوترهای کوچک
4. micro computer ریز کامپیوترها

Subject: Date:

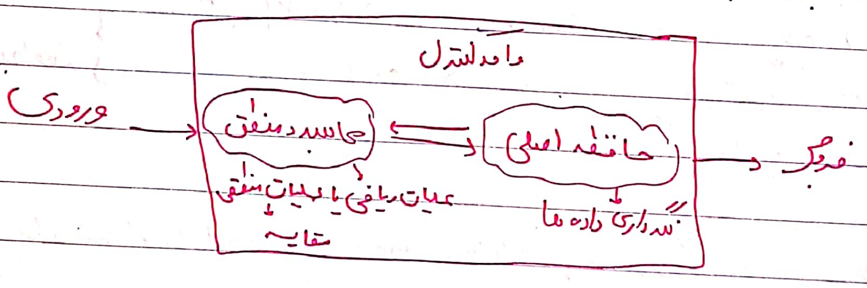
ابرکامپیوترها: گران ترین و قدرتمندترین رایانه ها سرعت بالایی دارند و بیشتر در زمینه های نظامی و تحقیقاتی استفاده می شود.

main frame: سرعت و قدرت بالایی دارند و در سازمان ها، دانشگاه ها استفاده می شود. توان محاسباتی این کامپیوترها نسبت به ابرکامپیوترها پایین تر است.

mini computer: با توجه به اینکه کامپیوترهای بزرگ قیمت بسیار بالایی داشتند این کامپیوترها وارد بازار شدند. توان محاسباتی کمتری دارند و توسط سازمان های کوچک تر استفاده می شوند.
این کامپیوترها قیمت پایین تر و حجم بسیار کوچکی دارند به همین دلیل مورد استقبال مردم و افراد عادی قرار دارند.

اجزای کامپیوتر به دو دسته سخت افزار و نرم افزار تقسیم می شوند.

سخت افزار: کلیه دستگاه های الکترونیکی و الکترومکانیکی (کیبورد)
نرم افزار: مجموعه برنامه هایی که برای کاربرد خاص نوشته می شود.



واحد محاسبه و منطق: واحدی است که تمام محاسبات و عملیات ریاضی مثل ضرب، جمع، تفریق، تقسیم را و منطقی مانند مقایسه دو مقدار در آن انجام می گیرد.

واحد کنترل: این بخش وظیفه نظارت و کنترل بر ورود داده ها از طریق ورودی و ذخیره آن ها در حافظه، انتقال اطلاعات از حافظه به واحد محاسبه و منطق و خروج اطلاعات از طریق واحد خروجی را دارد. به طور کلی وظیفه کنترل سایر بخش ها بر عهده دارد و تعیین می کند کدام عمل و در چه زمانی صورت پذیر است.

واحد محاسبه و منطق و واحد کنترل بخش مهم تشکیل دهنده واحد پردازش مرکزی یا cpu هستند.
اجزای دیگر cpu ثبات و حافظه نهان cache هستند.

Subject: Date:

ثبات ها حافظه های کوچک و بسیار سریعی هستند که داده ها برای پردازش و بعد از پردازش در هنگام انتقال به حافظه در ثبات ها قرار می گیرند.

حافظهٔ cache یک حافظه کوچک است که سرعت بالا دارد که بین حافظه اصلی و cpu قرار دارد.

سرعت دستیابی اطلاعات از حافظه cache نسبت به حافظهٔ اصلی بیش تر و نسبت به حافظه های جانبی (خیلی) بیشتر است.

سرعت: cache > حافظه اصلی > حافظه جانبی

داده هایی را که cpu از حافظهٔ اصلی فراخوانی می کنند پس از ارسال در cache قرار می گیرند که در فراخوانی های بعدی آن داده ها با سرعت بیشتری به cpu ارسال شوند.

محل نگهداری داده ها و برنامه ها به صورت موقت یا دائم واحد حافظه است.

با قطع برق میپره و پاک میشه — هم اطلاعات رو میتونیم هم برداشت کنیم بنویسیم و بخوانیم

واحد حافظه:
- حافظه اصلی main memory
  - حافظه با دستیابی تصادفی RAM Random access memory
  - حافظه فقط خواندنی ROM Read only memory (لازم برای اجرا)
- حافظه جانبی second memory

فقط قابل برداشت است و فقط خوانیم و فقط سیستم قابل اجرا میشود

که فقط توسط سازنده کارخانه انجام میشه تا برنامه ها اجرا بشود.

حافظه ها به دو دسته حافظه اصلی (main memory) یا حافظه جانبی (second memory) تقسیم می شود.

حافظه با دستیابی تصادفی قابل خواندن و نوشتن است و برای ذخیرهٔ اطلاعات کاربران استفاده می شود.

حافظه فقط خواندنی فقط قابل خواندن است و محتویات آن قابل تغییر نیست. این حافظه معمولاً توسط کارخانه سازنده پر می شود و حاوی دستورالعمل های برای راه اندازی اولیه کامپیوتر است.

حافظه جانبی نسبت به حافظه اصلی سرعت پایین تری دارند ولی حجم ذخیره سازی اطلاعات آن ها نسبت به حافظه اصلی زیاد است و اطلاعات ذخیره شده با قطع برق از بین نمی روند

CD فلش مموری Hard disk

Subject: Date:

مراحل حل مسئله: ۱- ارائه راهکارهای مختلف برای حل

۲. انتخاب بهترین راه حل

۳ نوشتن الگوریتم معادل آن.

٤ تست الگوریتم با داده های ورودی

۵ تبدیل الگوریتم به برنامهٔ کامپیوتری

۶ دادن برنامه به کامپیوتر و تست و اشکال‌گیری (اشکال‌زدایی)

۷- اجرای برنامه

الگوریتم باید به صورت دقیق بیان شود هر چند با زبان خودش.

قوانین الگوریتم: ۱- در ابتدا از کلمهٔ شروع و در پایان (انتها) از کلمهٔ پایان

۲ شماره‌گذاری همهٔ دستورات

۳ برای دریافت مقدار ~~ورودی~~ از کاربر کلمهٔ بخوان را استفاده می‌کنیم

٤ برای خروجی کلمهٔ چاپ کن

۵ حتماً باید متغیر تعریف کنیم

۶- برای انتصاب مقدار هم ← استفاده می‌شود.

* در انجام محاسبات ریاضی یا کار بر روی داده‌ها مکانی برای ذخیرهٔ داده‌ها و اضافه کردن نتایج در نظر گرفته می‌شود

که در کامپیوتر این مکان‌ها در خانه‌های حافظه ذخیره می‌شود برای اینکه بعداً بتوانیم به این خانه‌های حافظه مراجعه کنیم

به هر یک از آن‌ها یک نام می‌دهیم که به هر یک از این نام‌ها متغیر گفته می‌شود. نام متغیر به دلخواه انتخاب می‌شود

اما بهتر است نام آن متناسب با کاربرد باشد

* الگوریتمی بنویسید که ۳ عدد را از کاربر دریافت کند و میانگین‌ها را چاپ کند. ۱- شروع

۲- a، b، c را بخوان

۳- Sum ← a+b+c (هر چی اینجا اسم بذاری همو بنویس)

٤- Avrage ← S/۳

۵- arverge را چاپ کن

۶- پایان

جملات شرطی یا شرایط خاص

Subject: Date:

جملات شرطی: گاهی در الگوریتم نویسی با جملاتی که دارای شرط هستند روبه رو می شویم این جملات شرطی دستوراتی را بیان می کنند که در صورت برقراری آن شرط عملیات خاصی را اجرا کنند و در صورت عدم برقراری عملیات دیگری را انجام می دهند.

دو نوع جملهٔ شرطی وجود دارد: شرطی نوع ساده یا نوع اول و شرطی نوع دو

شرطی ساده: اگر (شرط) اجرای دستورات

شرطی نوع دو: اگر شرط برقرار باشد: دستورات ۱ اجرا
در غیر این صورت دستورات ۲ اجرا

* الگوریتمی بنویسید که ۲ عدد را گرفته و مجموع آن ها را چاپ کند. ۱- شروع
۲- a و b را بخوان
۳- $S \leftarrow a+b$
۴- S را چاپ کن
۵- پایان

الگوریتم بنویسید که طول و عرض مستطیل را بگیرد و محیط و مساحت مستطیل را محاسبه کند. ۱- شروع
۲- L و M را بخوان
۳- $P \leftarrow 2(M+L)$
۴- $S \leftarrow ML$
۵- S و P را چاپ کن
۶- پایان

الگوریتم

$30 \times 9 \times 300 \ldots \quad 11 \leftarrow 30$

دستمزد کارگری برای هر ساعت کار ۳۰ تومن است ساعت کارکرد در یک ماه را گرفته دستمزد ماهیانه را چاپ کند. ۱- شروع
۲- ساعت کارکرد را بخوان
۳- ساعت کارکرد × ۳۰ ← دستمزد ۱ ماه
۴- دستمزد ۱ ماه را چاپ کن
۵- پایان

boofi

Subject: Date:

۱- بیت: کوچکترین واحد حافظه [۰] [۱]

8 bit = 1 bite

۲- بایت

۱ بایت = ۸ بیت

۱۰۲۴ بایت = ۱ کیلوبایت 1KB

۱۰۲۴ کیلوبایت = ۱ مگابایت 1MB

۱۰۲۴ مگابایت = ۱ گیگابایت 1GB

* برای نشان دادن حروف و... از واحد کاراکتر استفاده می‌شود. caractor

۱ بایت برای هر یک کاراکتر. _ _ _ _ _ _ _ _

$۲^۸ = ۲۵۶$

که توی این خونه‌ها ۰ و ۱ می‌گذارند تا به ۶۸ برسند

۶۸ یعنی D مثلا

از ۰ تا ۲۵۵ حرف می‌توانیم داشته باشیم

از ۱۲۷ تا ۲۵۶ آزاد.

از ۰ تا ۱۲۷ را حروف و نمادها گذاشتند

بعد هر کشوری از ۱۲۷ به بعد می‌تونه

هر کشوری تعریف کنه.

برای ذخیره سازی حروف، ارقام و نمادها از واحد کاراکتر استفاده می‌شود.

برای ذخیره سازی کاراکترها به هرکدام از آن‌ها یک کد عددی نسبت داده می‌شود.

در گذشته پرکاربردترین کد ~~...~~ کد اصلی بود ASCII. اسکی که برای نمایش هر کاراکتر از یک بایت نوشته می‌شود. (استفاده شده)

از آن جا که هر بایت می‌تواند بین ۰ تا ۲۵۵ تغییر کند بنابراین تنها ۲۵۶ کاراکتر قابل تعریف است.

از این بین کدهای ۰ تا ۱۲۷ به صورت استاندارد برای علائم و حروف انگلیسی تعریف شده است.

کدهای بالاتر از ۱۲۷ برای هر کشور خالی گذاشته شده‌است تا بتوانند حروف خاص زبان خود را تعریف کنند.

تبدیل مبنا ۱۰ به ۲

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۰	۱	۰	۰	۰	۱	۰	۰

۶۸ = D

از کنار هم قرار گرفتن چند کاراکتر فیلد تشکیل می‌شود. در واقع گروهی از کاراکترها هستند که معنای خاصی دارند مانند نام یا نام خانوادگی

Ali — کاراکتر، کاراکتر، کاراکتر — فیلد

چند تا فیلد کنار هم تشکیل یک رکورد را می‌دهند.

رکورد کارمند: نام، نام خانوادگی، سن، ش ش — هرکدام جدا فیلد هستند.

Subject: Date:

چندتا رکورد کنار هم پیوسته: به مجموعه ای از رکوردهای مرتبط با هم (فایل) گفته می شود به عنوان مثال فایل کارمندان شامل چند رکورد کارمندان است.

به هر ۲ یا ۴ بایت کنار هم (کلمه) گفته می شود.

نرم افزارها برای ایجاد یا اجرای نرم افزارهای کاربردی نوشته می شوند.

نرم افزار:
- سیستمی:
  - سیستم عامل
  - مترجم ها ← کامپایلر، مفسر
  - برنامه های کمکی
- کاربردی: کاربر برای راه اندازی برنامه خودش انجام میده.

سیستم عامل: واسط بین سخت افزار و کاربر. وظیفه آن مدیریت منابع است. نوبت دهی پرینتر.

برنامه های کمکی مثل آنتی ویروس ها یا مدیریت فضای Disk

مترجم: کامپایلر خطاها را بررسی می کند اگر خطا نداشته باشد یک جا بررسی می کند و به زبان ماشین تبدیل می کند. هر دو با هم
ولی مفسر خط به خط چک می کند.

* خصوصیات الگوریتم ← باید ورودی و خروجی داشته باشد و جمله ها مفهوم باشد (زبان دقیق باشد) جزئیات کامل داشته باشد
مرحله به مرحله انجام شود.
دستورالعمل ها باید ترتیب داشته باشد. حتماً شروع خاتمه و پایان الگوریتم داشته باشیم.
برنامه بدون شروع و پایان اجرا نمی شود.

boofi

fizik olom computer کانال تلگرام

email: haghighi.w.tiau@gamil.com

Subject: Date:

وزن یک جسم را به کیلوگرم گرفته و به گرم نمایش دهد.

۱- شروع

۲- وزن جسم را به کیلوگرم بخوان

۳- وزن جسم $\times 10^3$ kg ← g وزن به

۴- وزن g را چاپ کن

۵- پایان

الگوریتم را بنویسید که وزن یک جسم را به گرم گرفته و مشخص کند چند کیلو و چند گرم است؟

۱- شروع

> [1,8] = 1kg
> 1×1000 = 1000
> 1550 − 1000 = 550 gr
> 1550 / 1000 → 1,5

۲- وزن جسم را به گرم بخوان (w)

۳- $[\frac{w}{1000}]$ ← G

۴- w − G×1000 ← m

۵- G و m را چاپ کن

۶- پایان

الگوریتمی بنویسید که عددی را دریافت کند و اگر عدد زوج بود کلمه زوج و اگر فرد بود کلمه فرد را در خروجی چاپ کند.

۱- شروع

> $\frac{28}{2} = 14$ $\frac{29}{2} = 14,5$

۲- A را بخوان

۳- $\frac{A}{2}$ ← B (قسمت صحیح تقسیم را در B می‌ریزیم)

> 29 − 2×14 = 1
> 28 − 2×14 = 0

۴- C ← A − 2*B

۵- اگر C برابر صفر بود (C=0) زوج را چاپ کن در غیر این صورت فرد را چاپ کن

۶- پایان

۳ عدد را دریافت کنید و مشخص کند که می‌توان با این ۳ عدد مثلث ساخت یا نه؟

۱- شروع

۲- x و y و z را بخوان

۳- اگر x+y > z بود

۴- اگر z+y > x بود

۵- اگر z + x > y بود برو به خط ۶ در غیر این صورت برو به خط ۷

۶- مثلث قابل ساخت است آن را چاپ کن

۷- مثلث قابل ساخت نیست آن را چاپ کن

۸- پایان

boofi

Subject: Date:

حلقهٔ تکرار: حتماً باید دارای شرط باشد وگرنه تا بی‌نهایت می‌رود.

حلقه‌های تکرار: گاهی اوقات در حل مسائل لازم است که عملیاتی چندین بار تکرار شود مثلاً قرار است الگوریتمی بنویسیم که ۱۰۰ عدد را دریافت کرده و جمع آن‌ها را محاسبه کنیم برای اینکه از صد بار نوشتن دستور عدد را بخوان و عددهای خوانده شده را جمع کن جلوگیری شود از حلقه‌های تکرار استفاده می‌کنیم حلقهٔ تکرار به گونه‌ای رفتار می‌کند که مجموعه‌ای از دستورات را تا زمانی که شرط خاصی برقرار باشد تکرار می‌کنیم

دو نوع حلقهٔ تکرار داریم ۱. حلقه‌های شرطی که شرط آن‌ها در ابتدای حلقه چک می‌شود دارای ساختار زیر هستند

تا زمانیکه (شرط) برقرار است مراحل a تا b را تکرار کن

a
⋮
b

حلقه‌های شرطی که شرط در انتهای حلقه چک می‌شود تکرار کن

a
⋮
b
اگر شرط برقرار است

تعداد اعداد دریافتی: i (شمارنده)

$i = 15$

تعداد کل اعداد شرط: n

$n = 50$

شرط حلقه: $i \leq n$

$i \leq n \Rightarrow 15 \leq 50$ ✓

$Sum \leftarrow Sum + i$

Subject: Date:

فلوچارت: ۱- شروع و پایان (begin / end)

۲- برای دستورات محاسبات و انتساب ها

۳- برای ورودی گاهی برای خروجی

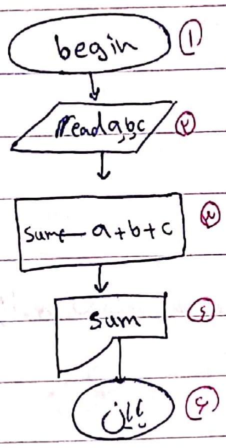
۴- برای شرط ها (if)

۵- برای خروجی همیشه

برای چاپ کن ← write

بخوان ← read

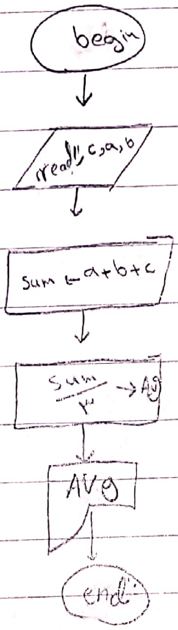
اگر بخواهیم از متوازی الاضلاع استفاده کنیم باید توش write بنویسیم (write a,b) را از قسمت ۵ (a,b)



۱- شروع
۲- a و b و c را بخوان
۳- $sum \leftarrow a+b+c$
۴- sum را چاپ کن
۵- پایان

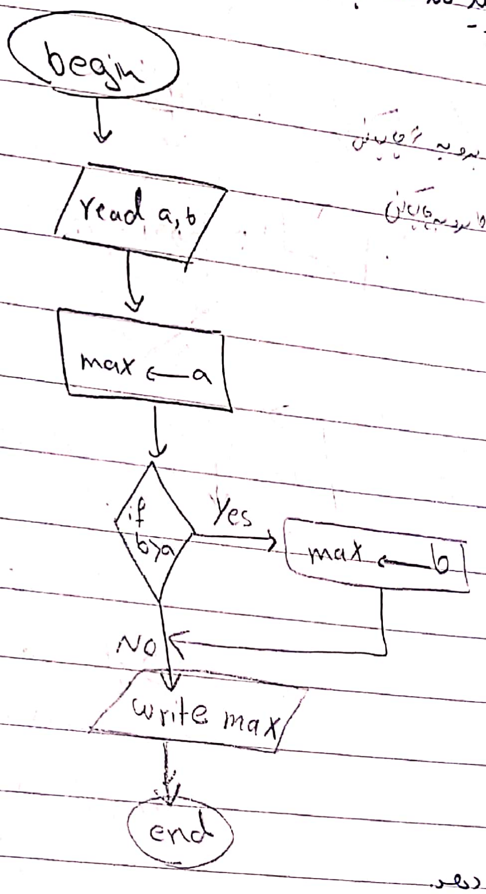
فلوچارت مربوط به میانگین گیری ۳ عدد را بکشید

۱- شروع
۲- a و b و c را بخوان
۳- $sum \leftarrow a+b+c$
۴- $Aveg \leftarrow \frac{sum}{۳}$ (میانگین)
۵- پایان



Subject: Date:

فلوچارتی رسم کنید که ۲ عدد از ورودی دریافت کرده بزرگ‌ترین آن‌ها را پیدا کند و در خروجی چاپ کند.



۱- شروع

۲- a و b را بخوان

۳- max ← a

۴- اگر a<b باشد max ← b

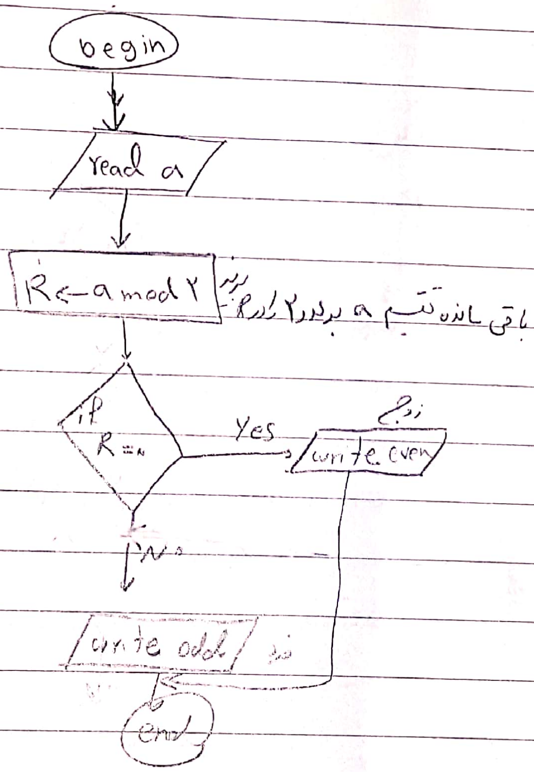
۵- در غیر این صورت (a ماکزیمم بود) برو به دستور ۶

۶- max را چاپ کن

۷- پایان

فلوچارتی رسم کنید که عددی را از ورودی دریافت کند و فرد یا زوج بودن آن را تشخیص دهد.

mod باقی مانده



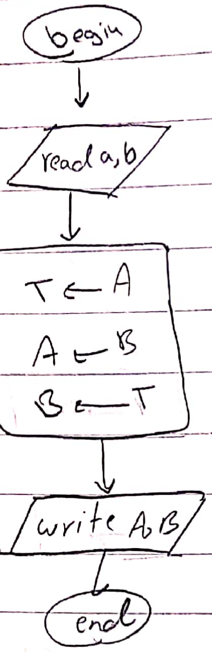
Subject: Date:

① فلوچارتی رسم کنید که ۲ عدد به عنوان a و b را دریافت کند و سپس مقادیر آن ها را با یکدیگر جابجا کنند. راه حل دیگر؟

$A = 10$

$B = 5$

$T =$ متغیر کمکی



۱. شروع

۲. a و b را بخوان

۳. $T \leftarrow A$ الان A خالیه

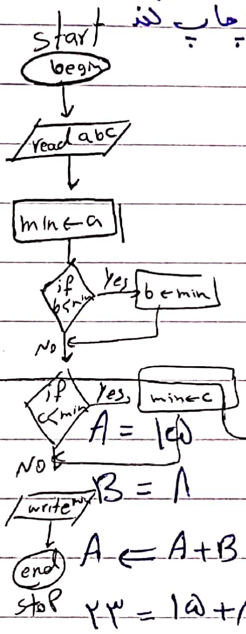
۴. $A \leftarrow B$ الان B خالیه

۵. $B \leftarrow T$

فلوچارتی رسم کنید که ۳ عدد را از ورودی دریافت کند کوچک ترین عدد را یافته در خروجی چاپ کند

۱. شروع

۲. a و b و c را بخوان



$A = 15$

$B = 8$

$A \leftarrow A+B$

$23 = 15+8 \} \rightarrow A = 23$

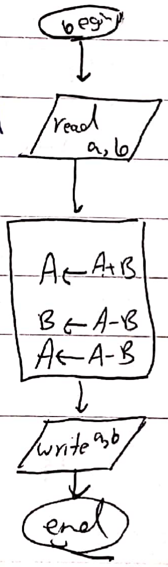
$B \leftarrow A-B$

$15 \leftarrow 23-8 \} \rightarrow B = 15$

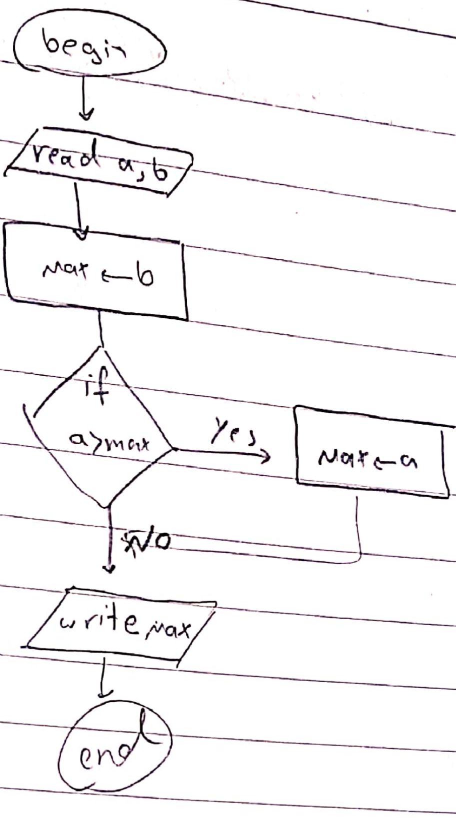
$A \leftarrow A-B$

$8 \leftarrow 23-15 \} \Rightarrow A = 8$

تمرین ① راه دیگرش



boofi



دو عدد دریافت، بزرگترین انتخاب کنید

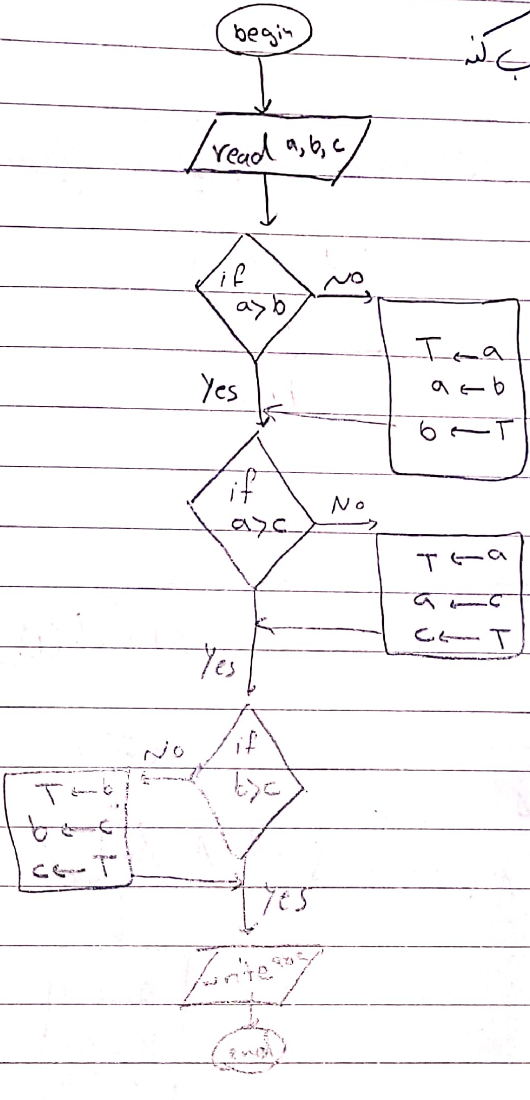
۱- شروع

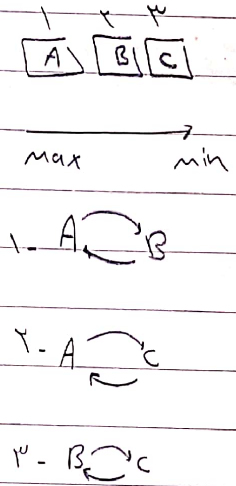
۲- a , b را دریافت کند

۳- max ← b

۴- اگر a > b

الگوریتمی بنویسید که ۳ عدد را خوانده و آن ها را به ترتیب نزولی مرتب کند





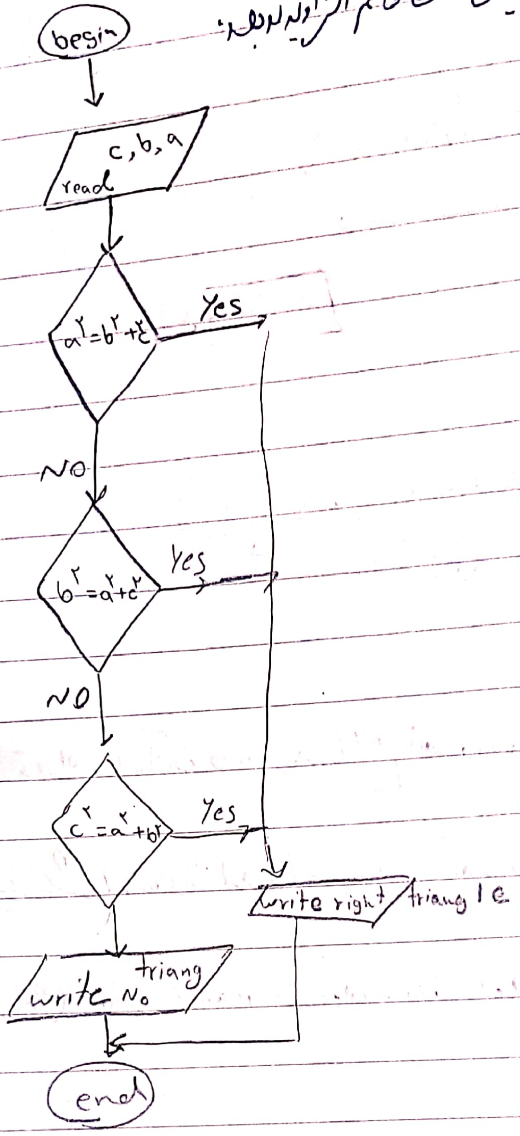
Subject: Date:

...ورودی که ۳ باید تشکیل مثلث قائم الزاویه را بدهد:

$a^2 = b^2 + c^2$

$b^2 = a^2 + c^2$

$c^2 = a^2 + b^2$



اگر توی چند خط باشند.

/* .... */ ← دستور برنامه نیست، برنامه‌نویس فهم خودش یادداشت یا دستیاران نوشته

Subject: Date:

```
// this Program is c++
```
← این قسمت توسط کامپایلر به زبان ماشین ترجمه نمی‌شود.

طرح کلی برنامه در c++

← number sign است

شروع می‌شود

```
#include <iostream>

void main
{
    cout << "welcome to c++";
}
```

#include <iostream> ← دستورات پیش‌پردازنده هستند، که دستوری c++ نیست، کامپایلر قبل از اینکه دستورات برنامه را کامپایل کند (ترجمه به زبان ماشین) کند دستورات پیش‌پردازنده را که با علامت # شروع می‌شود کامپایل می‌کند.

② برای تعریف زیربرنامه‌ها و توابعی که قصد استفاده از آن‌ها در برنامه داریم استفاده می‌شود.

③ زمانی که قرار است توی خروجی چیزی چاپ شود یا داده‌ای از طریق کاربر یا صفحه کلید دریافت شود

قسمت اصلی برنامه (دستورات اصلی برنامه‌ها) در قسمت main قرار می‌گیرند. در ابتدا و انتهای دستوراتی که درون main است باید با آکولاد مشخص شود.

در انتهای هر خط دستورات اصلی باید علامت ; حتما باشد semi colon

④ خروجی: زمانی که قرار است به کامپیوتر دستوری جهت انجام داده شود. برای اینکه بهتر بود نشون می‌دهیم اینجوری کار کنیم.

Subject .... Date ....

مبانی کامپیوتر

نوع داده های اصلی :

char	کاراکتر	۸ بیت
int	اعداد صحیح	۱۶ بیت
float	اعداد اعشاری	۳۲ بیت
double	اعداد اعشاری با دقت مضاعف	۶۴ بیت

ـ ارقام ۰ تا ۹ می توانند در نام متغیر استفاده شوند و همچنین آندرلاین (_)

شناسه ها:

1umberX => umber1 ✓

۱. کاراکتر A تا Z ۲. ارقام ۰ ... ۹ (نمی توانند در ابتدا بیاید) ۳. علامت زیر خط ـ به عنوان فاصله استفاده می شود.

number one X فاصله نمی توانیم داشته باشیم

numberOne ✓

number_one ✓

شناسه نامی است که به متغیر یا توابع دیگر برنامه داده می شود که برای آن می توان از این ۳ مورد بالا استفاده کرد.

ـ زبان ++c به حروف کوچک و بزرگ حساس است

ـ شناسه با رقم نباید آغاز شود. (۲)

ـ در اسامی دو قسمتی گذاشتن فاصله اشتباه است. (۳)

* تعریف نوع داده : متغیر مکانی از حافظه است که مقداری در آن ذخیره می شود هر متغیر قبل از استفاده در برنامه باید اعلان شود یعنی نوع آن باید مشخص شود.

نوع داده	نام داده	
int	number	;
float (مثل معدل)	average	;

int number = 12;

Subject .... Date ....

cout << "welcome to c++ \n" ;

\n : انتقال به خط بعدی

\n	انتقال به خط بعدی
\t	انتقال به اندازه tab
\a	صدای بوق
\\	یک \ چاپ می شود

عملگر :

(علامت منفی مثبت) یکانی : ۳ ، -a ← قبل حروف قرار می گیرد.

(بین دو متغیر) دودویی : a + b

عملگر های اساسی : منفی کردن - یک عملوند دارد

+	a+b	
-	a-b	
* ضرب	a*b	
/ تقسیم ← اگر دوتا عدد که هردو صحیح باشد تقسیم صحیح به صحیح انجام می شود ولی اگر یکی از آن ها اعشاری باشد تقسیم اعشاری است.	a/b	نمایش خارج قسمت
% باقی مانده تقسیم ← فقط برای اعداد صحیح استفاده می شود	a%b	نمایش باقی مانده

در تقسیم a/b : درصورتی که دو عملوند صحیح باشند تقسیم به صورت صحیح به صحیح است و اگر یکی از عملوندها اعشاری باشند

تقسیم به صورت اعشاری است

برای اولویت ها : منفی ساز - (۱

پرانتز (۲

% ، / ، * (۳ از سمت چپ به راست انجام می شود

+ ، - (۴

- اگر چند پرانتز داشته باشیم ابتدا داخلی ترین پرانتز محاسبه می شود.

- درصورتی که عملگرهای هم ارزش با اولویت یکسان داشته باشیم از چپ به راست اولویت گذاری و ارزیابی می شود.

Subject .... Date ....

- اولویت بندی؟

مثال $z = x * p \% (y + m / (n - L))$

(اولویت اول ①، چون داخل پرانتز است ③ ②؛ پرانتز دوری خودش عملگر دارد؛ ④ ⑤؛ یک نوع عملگر انتسابی (ساده‌ترین عملگر انتساب) ⑥)

انتساب: همهٔ عملیات‌های راست را داخل چپ می‌ریزد.

عملگرهای انتساب: ابتدا یه مقدار a اضافه شود سپس استفاده در برنامه شود ← $++a$

انتسابی یکانی:
- $++a$ ← ابتدا یه مقدار a اضافه شود سپس در برنامه استفاده شود
- $a++$ ← ابتدا مقدار فعلی a استفاده شود سپس مقدار a افزایش یابد
- $--a$ ← ″ ″ کم ″ ″
- $a--$ ← ″ ″ ″ کم ″ ″ ″ ″

int a = 4;

$++a \Rightarrow a = 5$

عملگر انتساب دوروی:

+=	a += 4	a = a + 4
-=	a -= 4	a = a - 4
*=	a × = 4	a = a × 4
/=	a /= 4	a = a / 4
%=	a %= 4	a = a % 4

عدد ۴ به عنوان مثال است

عملگرهای مقایسه‌ای:
- <= ← اولین: a و b مقایسه می‌شوند در نهایت جواب صحیح را می‌دهد.
- >=
- == این دو مقدار یا عملوند باهم برابر هستند ولی = یعنی مقدار سمت راست می‌ریزد توی چپ
- != نامساوی
- <
- >

جواب یا نتیجهٔ عملگرهای مقایسه‌ای True یا False خواهد بود

اولویت ها: ۱) <= ، >= ، < ، > از چپ به راست محاسبه می‌شود اگر عملگرهای کنار هم باشند

۲) == ، !=

Subject AND Date

AND &&
OR ||
NOT !

عملگرهای منطقی : not, or , and

اولویت :
1) !
2) &&
3) ||

* مقداری که در خروجی عملگرهای منطقی نشان داده می شود عدد صفر یا یک است که یک به معنای درست و صفر به معنای غلط است.

a	b	And	OR	NOT	&&	NOT OR
0	0	0	0	1		1
0	1	0	1	1		0
1	0	0	1	1		0
1	1	1	1	0		0

عملگر شرطی :
شرط ? عبارت ۱ : عبارت ۲
(اگر درست بود — عبارت ۱ ، اگر درست نبود — عبارت ۲)

a = (K<10) ? 100 : 70
T F

در صورت درست بودن شرط عبارت ① و در غیر این صورت عبارت ۲ بازگردانده می شود.

معادل:

```
if K<10   a=100
else      a=70
```

```
cout << "Please enter number";
```

② cout << a << b => cout << متغیر اول << متغیر دوم

① * cin >> a >> b => cin >> متغیر اول >> متغیر دوم
(a: متغیر اول ، b: متغیر دوم)

دستور cin برای دریافت اطلاعات از صفحه کلید استفاده می شود. فرمت این دستور به شکل ① *

دستور cout برای نشان دادن نتایج در خروجی از دستور cout استفاده می شود که فرمت آن به شکل زیر است ②

در این شکل مقادیر خروجی a و b نشان داده می شود ولی اگر جمله ای بین " " نوشته شود عین آن عبارت است.

خروجی نمایش داده می شود — برای دریافت عدد صحیح و اعشاری

```
int x
float f
cin >> x >> f;
```

برخی فایل های سرآیند که باید به همراه #include بیایند

1) iostream.h → cout, cin برای توابعی مثل

۲) conio.h → gotoxy, clrscr, getch → برای گرفتن character (داخل < > آید)
clrscr ↓ clearscreen

۳) string.h → strcpy

٤) graphics.h → line, circle برای کشیدن خط و دایره

۵) math.h → برای استفاده از توابع ریاضی → sqrt, sin, tan, ...
sqrt: جذر
فرمول های آماری — کتابخانه های آماری

* ۱. شامل توابع خروجی ورودی مانند cout, cin می شود

۲. شامل توابعی برای پاک کردن صفحه نمایش clrscr، برای انتقال مکان نما gotoxy

برای دریافت یک کاراکتر از ورودی getch استفاده می شود

۳. شامل توابع کار با رشته ها مانند strcpy برای کپی استفاده می شود و ... استفاده می شود

٤. شامل توابع برای کشیدن یا رسم کردن خط (line) و یا کشیدن دایره مانند circle و ... است

۵. شامل توابع ریاضی مانند sin, tan, sqrt جذر و ... است

* ساختار کلی کدی که یک کاراکتر از کاربر دریافت می کند

```
#include <conio.h>
void main
{
  getch();
}
```

* برنامه ای بنویسید که دو عدد صحیح را خوانده و در خروجی نمایش دهد

متغیر → a, b اینا کاراکترند

```
#include<iostream.h>
#include <conio.h>
void main
{
  int a,b;
```

cout

صفحه بعدی

Subject .... Date ....

```
# include <iostream.h>
# include <conio.h>
voidmain
{
int a, b ;
cout << "Please enter twonumber";
cin >> a >> b
cout << a + << b ;
getch( );
}
```

متغیرها (int a, b)

getch ← زمانیکه از
کاراکتر استفاده
کنیم باید از getch
استفاده کنیم

Subject .... Date

برنامه‌ای بنویسید که دو عدد را خوانده و مجموع آن‌ها را نمایش دهد. به متغیر سوم نیاز داریم

```
# include <iostream.h>
# include <conio.h>
  void main
  {
    int a, b, P;
cout << "Please enter two umbers";   [ 2 numbers ]
    cin >>a >>b
P = a+b;
  cout << a \t << b \t << P;
یا
cout << a \t << "+" << b \t << "=" << P;
  getch();
  }
```

فرض: ۱) ۱۱ tab ۷ tab ۴   a = ۴, b = ۷

۲) ۴ + ۷ = ۱۱

برنامه‌ای بنویسید که اضلاع مستطیل را دریافت کند مساحت آن را حساب کند.

a = طول
b = عرض
P = مساحت
g = محیط

```
# include <iostream.h>
# include <conio.h>
viod main
{
  int a, b, P, g;
  cin >> a >> b;
  P = a×b;
  g = (a+b)×۲;
  cout << a \t << b \t << P <<
  getch();
}
```

فرض:
a = ۴
b = ۳
۴ ۳ ۱۲ ۱۴

Subject .... Date ....

برنامه‌ای بنویسید که ۲ عدد را خوانده و عدد بزرگ‌تر را نشان دهد.

```
# include <iostream.h>

# include <conio.h>
  void main
{
   int a,b, max;
     cin>>a>>b;
   max= (a>b) ?   a:b;

 cout<<"max ="<<max;

 getch ( );

   }
```

به جای این: if(a>b)
max =a;
else
max = b;

اگر a>b بود مقدار a را می‌دهد ماکس اگر نبود b را بگذار

a = ۱۵
b = ۲۸

فرض:

max = ۲۸

```
if (شرط)
    دستور ;
```

اگر شرط درست بود انجام میده
اگر نبود میره بعدی

```
if (شرط)
   دستور ۱ ;
     else
   دستور ۲ ;
   else
     ⋮
   else
   دستور آخر
```

می‌تواند تعداد دستورات بیشتر از ۲، ۳ تا باشد

(if) باید باشد

clips TM

Date

برنامه ای که عددی را خواندن و تشخیص دهد عدد (+) است یا (-)

```
#include<iostream.h>
# include<conio.h>
void main
{
int num;
cout<<"Please enter anumber:";
cin>>num;
if (num>=0)
{
cout<<num<<"is Positive";
}
else
{
cout<<num<<"is negative";
}
getch();
}
```

متغیر	هدف
num	عدد خوانده شده

خروجی : Plese enter anumber

مثلاً عدد 8 را وارد می کنیم isPositive 8

Subject .... Date ....

برنامه‌ای بنویسید که دو عدد را خوانده تعیین کند که باقی مانده تقسیم عدد اول بر عدد دوم زوج است یا فرد؟

```
#include<iostream.h>
#include <conio.h>
voidmain
{
int a,b;
cout<<"Please enter two number:";
cin>>a>>b
if (a%b)%2 ==0
{
cout<<a<<"mod"<<b<<"is even";
}
else
{
cout<<a<<"mod"<<b<<"is odd";
}
getch();
}
```

برنامه‌ای بنویسید که مقدار x را خوانده قدر مطلق آن را نمایش دهد؟

$$|x| = \begin{cases} -x & \text{اگر } x<0 \text{ باشد} \\ 0 & \text{اگر } x=0 \text{ باشد} \\ x & \text{اگر } x>0 \text{ باشد} \end{cases}$$

قدر مطلق

```
if ( عبارت شرطی ) ;
{
  دستورات ;
}
else if ( عبارت شرطی ۲ )
{
  دستورات ۲ ;
}
  :
else
{
  دستورات n
}
```

```
#include <iostream.h>
#include<conio.h>
voidmain
{
int x, abs;
cout<<"Please enter anumber:";
cin>> x
if(x<0)
{
abs=-x;
}
```

```
else if (n == 0)
{
abs = 0;
}
else
{
abs = n;
}
cout << "abs" << n << " = " << abs;
getch();
}
```

← آخرین شرط در if است

	متغیر
عدد خوانده شده	n
قدر مطلق	abs

اگر تعداد if های تو در تو زیاد شود از دستور سوییچ استفاده می‌شود.

هر عبارت داخل پرانتز برابر با مقدار این کیس (case) باشد دستورات آن اجرا می‌شود و دستور break ساختار سوییچ را خاتمه می‌دهد.

```
switch ( عبارت ) {
case مقدار① :
دستورات ;
break;
case مقدار۲ ;
دستورات۲ ;
break ;
   ⋮
case مقدار n
دستورات n
break;
default;
دستورات n+1
}
```

برنامه‌ای بنویسید که دو عدد و یک عملگر را دریافت کرده و حاصل عبارت را نمایش دهد.

clips™

Subject .... Date ....

```
#include <iostream.h>
#include <conio.h>
voidmain()
{
  float a,b,c;
  char ch;
cout<<"Please enter number and oPand another num:";
switch (ch)
{
 case'+':
 c=a+b;
 break;
cas'-':
c=a-b;
break;
cas'*':
c=a*b;
break;
cas'/':
c=a/b;
break;
default:
cout<<"Invalid operator";
 getch();
 return;
 }
 cout<<"result is"<<c;
    get();
  }
```

(عملگر written above "and")

a عدد اول

ch کاراکتر (عملگر)

b عدد دوم

c حاصل عبارت

خروجی: please ...... ۱۲×۳

result is ۳۶

مثال۱۶ – ۱. **الگوریتم و فلوچارتی که ابتدا عدد صحیح و مثبت n را از ورودی خوانده، سپس معدل n دانشجو را می‌خواند و میانگین معدل آن‌ها را محاسبه کرده و در خروجی می‌نویسد.**

داده: n (تعداد معدل)

خواسته: ave (میانگین معدل دانشجویان)

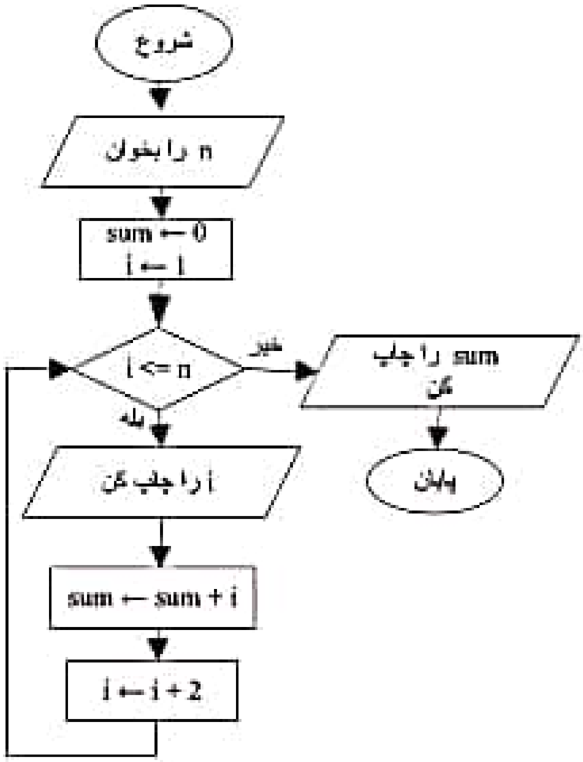
مثال ۱۱ – ۱. الگوریتم و فلوچارتی که عددی مثبت را خوانده، اعداد فرد ۱ تا عدد خوانده شده (n) و مجموع آن‌ها را نمایش می‌دهد.

توضیح: در این الگوریتم باید مجموع اعداد فرد نیز نمایش داده شود. برای این منظور، یک ظرف به نام sum در نظر گرفته می‌شود که در ابتدا صفر در آن قرار می‌گیرد. یعنی مجموع اعداد فرد در ابتدا صفر است.

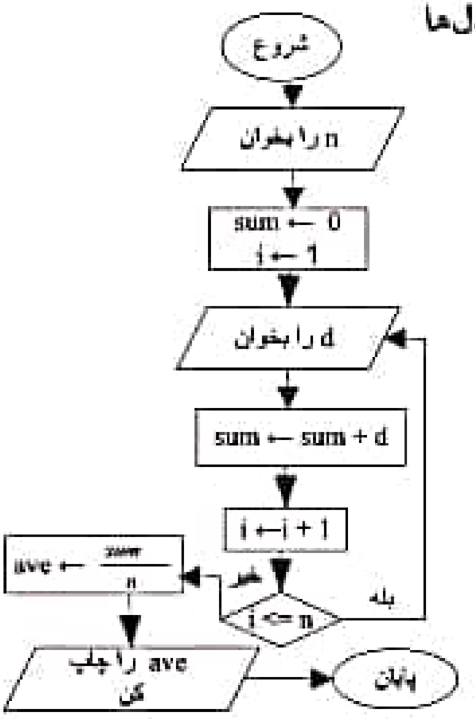
داده : n (عدد خوانده شده)

خواسته: نمایش اعداد فرد ۱ تا n و مجموع آن‌ها

رابطه: ایجاد یک حلقه و استفاده از متغیر sum برای نگهداری مجموع



۱. شروع
۲. n را بخوان
۳. صفر را در sum قرار بده
۴. یک را در i قرار بده
۵. تا وقتی که i <= n تکرار کن
الف: i را چاپ کن
ب: i را با sum جمع کرده، در sum قرار بده
ج: i + 2 را در i قرار بده
۶. sum را چاپ کن
۷. پایان



رابطه:

میانگین معدل = مجموع معدل‌ها تقسیم بر تعداد معدل‌ها

مجموع معدل‌ها :sum

شمارنده : i

$i \leftarrow i + 1$

معدل خوانده شده : d

$sum \leftarrow sum + d$

۱. شروع

۲. n را بخوان

۳. $sum \leftarrow 0$، $i \leftarrow 1$

٤. d را بخوان

٥. $sum \leftarrow sum + d$

٦. $i \leftarrow i + 1$

۷. اگر i <= n بود برو به دستور ۴

۸. $ave \leftarrow \frac{sum}{n}$

۹. ave را چاپ کن

۱۰. پایان

این الگوریتم ابتدا تعداد دانشجویان (n) را می‌خواند. سپس، مجموع معدل‌ها (sum) را برابر صفر قرار می‌دهد و یک واحد شمارنده (i) قرار می‌دهد. یک نمره را خوانده، این معدل را با مجموع معدل‌ها جمع کرده در آن قرار می‌دهد. سپس به شمارنده یک واحد اضافه می‌کند. اکنون اگر شمارنده کوچک‌تر یا مساوی تعداد معدل‌ها (n) باشد، معدل بعدی را می‌خواند، وگرنه مجموع معدل‌ها (sum) تقسیم بر تعداد معدل‌ها (n) را در میانگین معدل‌ها (avg) قرار می‌دهد و آن را چاپ می‌کند.

مثال۱۷ - ۱. الگوریتم و فلوچارتی که ابتدا تعداد دانشجویان یک کلاس (عدد صحیح و مثبت n) را از ورودی می‌خواند و سپس، نمرات آن دانشجویان را خوانده و بیشترین و کم‌ترین نمره کلاس را پیدا کرده و در خروجی نمایش می‌دهد (n >= 2) .

تعداد دانشجویان : n ، نمره هر دانشجو: x

کم‌ترین نمره : min ، بیشترین نمره : max

۱. شروع

۲. n را بخوان

۳. $i \leftarrow 2$

۴. x را بخوان

۵. $min \leftarrow x$ و $max \leftarrow x$

۶. x دیگری را بخوان

٧. اگر x < min آنگاه min ← x

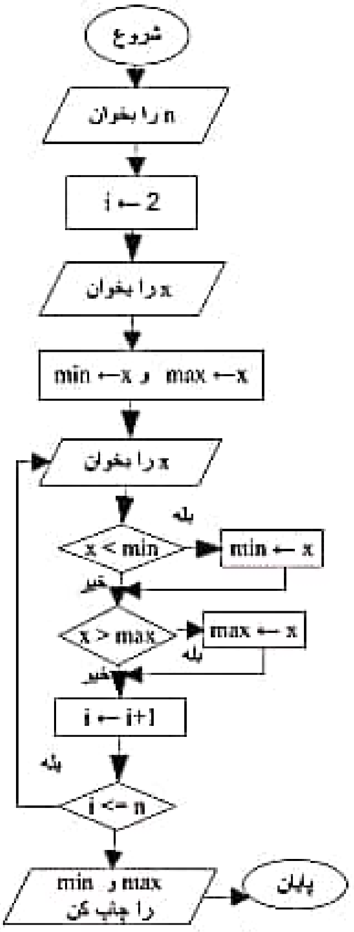
٨. اگر x > max آنگاه max ← x

٩. i ← i + 1

١٠. اگر i <= n برو به دستور ۶

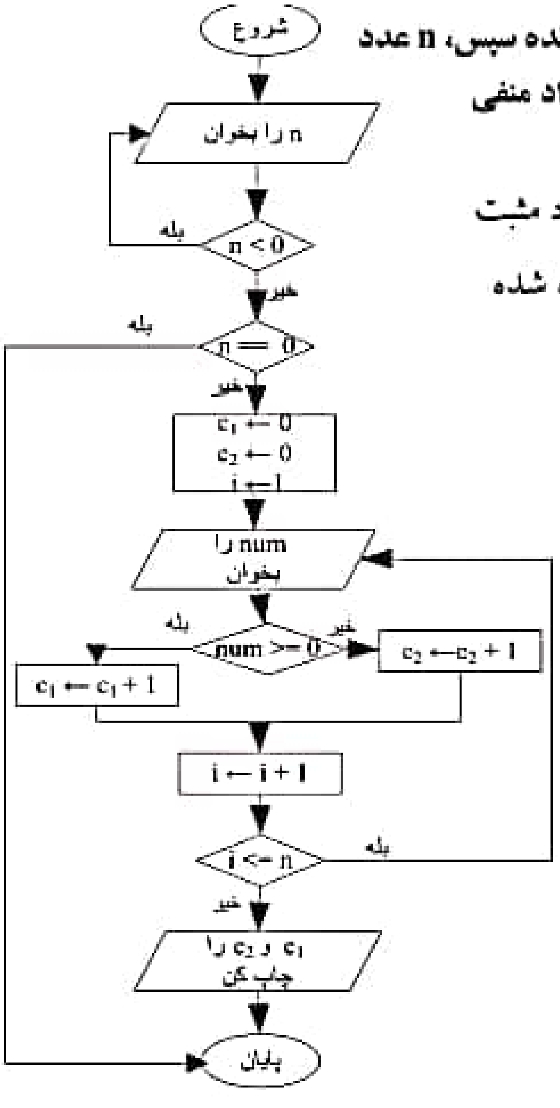
١١. min و max را چاپ کن

١٢. پایان



این الگوریتم ابتدا تعداد دانشجویان را خوانده، سپس در مقدار ۲ را درشمارنده قرار می‌دهد. چون ابتدا یک نمره (x) را خوانده در کم‌ترین نمره (min) و بیشترین نمره (max) قرار می‌دهد. سپس، یک نمره دیگر می‌خواند (دومین نمره را می‌خواند سپس i باید قبل از شروع حلقه ۲ باشد). اگر این نمره خوانده شد کم‌تر از کم‌ترین نمره باشد، آن را در کم‌ترین نمره (min) قرار می‌دهد. سپس نمره را با بیشترین نمره مقایسه می کند. اگر این نمره بیشتر از بیشترین نمره مقایسه می کند. اگر این نمره بیشتر از بیشترین نمره باشد آن را در بیشترین نمره قرار می‌دهد. در ادامه شمارنده را یک واحد اضافه می کند. اگر شمارنده کوچک‌تر یا مساوی n باشد، یک نمره دیگر می‌خواند و این روند را ادامه می‌دهد. و گرنه کم‌ترین و بیشترین نمره را چاپ می کند.

مثال ۱۸ - ۱. الگوریتم و فلوچارتی که ابتدا n را از ورودی خوانده سپس، n عدد دیگر را نیز می‌خواند و از بین آن‌ها تعداد اعداد مثبت و تعداد اعداد منفی را تعیین کرده و در خروجی می‌نویسد.



n: تعداد اعدادی را که قرار است خوانده شود، c_1: شمارشگر اعداد مثبت

c_2: شمارشگر اعداد منفی، num : اعداد خوانده شده

i: شمارشگر تعداد اعداد خوانده شده

۱. شروع

۲. n را بخوان

۳. اگر $n < 0$ برو به ۲

٤. اگر $n == 0$ برو به مرحله ۱۱

۵. $c_1 \leftarrow 0$ و $c_2 \leftarrow 0$ و $i \leftarrow 1$

٦. num را بخوان

۷. اگر $num >= 0$ آنگاه $c_1 \leftarrow c_1 + 1$

در غیر این صورت $c_2 \leftarrow c_2 + 1$

۸. $i \leftarrow i + 1$

۹. اگر $i <= n$ برو به ۶

۱۰. c_1 و c_2 را چاپ کن

۱۱. پایان

این الگوریتم ابتدا n را می‌خواند. اگر n کوچک‌تر از صفر باشد، عدد دیگری را می‌خواند. اما، اگر n برابر صفر باشد، الگوریتم خاتمه می‌یابد. وگرنه، مقادیر c1(تعداد اعداد مثبت) و c2 (تعداد اعداد منفی) را برابر صفر قرار می‌دهد. چون در ابتدا تعداد اعداد مثبت و منفی باید صفر باشد. سپس، یک را در شمارنده (i) قرار می‌دهد. در ادامه یک عدد (num) را خوانده، اگر عدد بزرگ‌تر یا مساوی صفر باشد، به تعداد اعداد مثبت یک واحد اضافه می‌کند ($c1 \leftarrow c1 + 1$). در پایان به شمارنده یک واحد اضافه می‌کند. اگر i (شمارنده) کوچک‌تر یا مساوی n باشد، عدد بعدی را خواهد خوانده، این روند ادامه می‌یابد. وگرنه، c1 و c2 را چاپ می‌کند.

اکنون اگر i کوچک‌تر یا مساوی n باشد، عددی دیگر را خوانده و این روند را ادامه می‌دهد. وگرنه، count را چاپ می‌کند.

نکته:	اگر اپراتور % بین دو عدد صحیح قرار گیرد نتیجه باقی مانده تقسیم صحیح خواهد بود.

مثال ۲۳ -۱. الگوریتم و فلوچارتی که ابتدا تعداد دانشجویان یک کلاس را از ورودی می‌خواند(n)، سپس نمرات آن دانشجویان را به ترتیب از ورودی خوانده، نمرات اول و دوم کلاس را تعیین کرده و در خروجی می‌نویسد.

داده
- تعداد: n
- نمرات: x, y

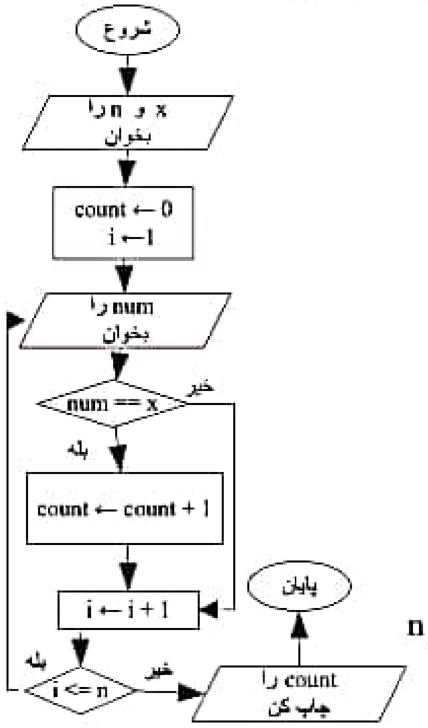
خواسته
- نمره اول: max1
- نمره دوم: max2
- k = 1, 2 , ..., n

رابطه: مقایسه

۱. شروع

۲. n را بخوان

مثال ۱۹ – ۱. الگوریتم و فلوچارتی که ابتدا اعداد x و n را از ورودی می‌خواند، سپس n عدد دیگر را نیز خوانده، تعداد دفعات تکرار عدد x، در آن n عدد را تعیین کرده و در خروجی چاپ می‌کند.



۱. شروع

۲. x و n را بخوان

۳. count ← 0 و i ← 1

٤. num را بخوان

۵. اگر num == x باشد آنگاه count ← count + 1

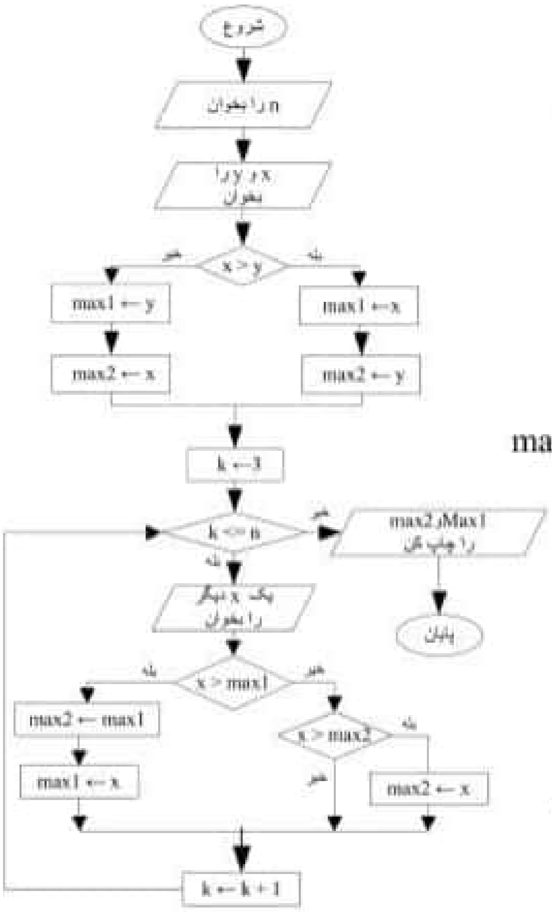
٦. i ← i + 1

۷. اگر i <= n آنگاه برو به مرحله ۴

۸. count را چاپ کن

۹. پایان

این الگوریتم ابتدا x (عددی که تعداد تکرار آن باید تعیین شود) و n (تعداد اعدادی که باید خوانده شود) را می‌خواند. سپس count (تعداد تکرار) را برابر صفر و i (شمارنده حلقه) را برابر یک قرار می‌دهد. در ادامه عـددی را خوانـده، در num قـرار می‌دهد. اگر x برابر num باشد، به count یک واحد اضافه می‌کند، شمارنده را یک واحد اضافه می‌نماید (i = i + 1).



۳. x و y را بخوان

۴. اگر $x > y$ باشد آنگاه $max1 \leftarrow x$, $max2 \leftarrow y$

وگرنه، $max1 \leftarrow y$, $max2 \leftarrow x$

۵. $k \leftarrow 3$

۶. تا هنگامی که $k <= n$ تکرار کن

الف: یک x دیگر را بخوان

ب: اگر $x > max1$ آنگاه $max2 \leftarrow max1$, $max1 \leftarrow x$

وگرنه، اگر $x > max2$ آنگاه $max2 \leftarrow x$

ج: $k \leftarrow k + 1$

۷. max1 و max2 را چاپ کن

۸. پایان

این الگوریتم ابتدا تعداد دانشجویان (n) را می‌خواند. سپس دو نمره x و y را خوانده، اگر x بزرگ‌تر از y باشد، x را در max1 (نمره اول کلاس) و y را در max2 (نمره دو کلاس) قرار می‌دهد. وگرنه y را در max1 و x را در max2 قرار خواهد داد. اکنون شمارنده k را برابر ۳ قرار می‌دهد. چون سومین نمره را می‌خواهد بخواهند (دو نمره x و y را قبلاً خوانده است). اکنون تا زمان که k کوچک‌تر یا مساوی n باشد (تعداد نمرات خوانده شده، کوچک‌تر یا مساوی تعداد دانشجویان باشد)، اعمال زیر را انجام می‌دهد:

- نمره دیگری (x) را می‌خواند.

- نمره دیگری (x) را می‌خواند.
- اگر x بزرگ‌تر از max1 باشد (یعنی x بزرگ‌تر از اولین نمره است). ابتدا، اولین نمره قبلی (مقدار فعلی max1) را در max2 قرار می‌دهد و x را در max1 قرار می‌دهد تا x به عنوان اولین نمره انتخاب شود. وگرنه، اگر x بزرگ‌تر از max2 باشد، x را در max2 قرار می‌دهد.
- به شمارنده k یک واحد اضافه می‌کند.
- به محض این که شمارنده k بزرگ‌تر از n گردید، max1 و max2 را نمایش می‌دهد.

مثال ۸ – ۲. قطعه برنامه زیر را اصلاح کنید:

```
int main()
{
  int num;
  cout >> "Enter a number:;
  cin >> num;
  cout >> " Number is" >> num >> endl;
```

این برنامه به صورت زیر تغییر می‌یابد.

```
#include <iostream.h>
int main()
{
  int num;
  cout <<"Enter a number:";
  cin >> num;
  cout << " Number is " << num << endl;
 return 0;
}
```

مثال ۴ – ۲. چه اشتباهی در برنامه زیر وجود دارد؟

```
#include <iostream>
int main()
{
   cout << "hello, world\n"
   return 0;
}
```

در خط اول فایل <iostream> باید به <iostream.h> تغییر یابد و انتهای دستور cout باید کاراکتر ; قرار بگیرد.

مثال ۵ – ۲. چه اشتباهی در توضیح زیر وجود دارد؟

```
cout << "hello, /*change ?*/ world.\n"
```

توضیحات نباید در بین جفت کوتیشن (") قرار بگیرند.

مثال ۶ – ۲. چه اشتباهی در برنامه زیر وجود دارد؟

```
#include < iostream >
int main()
{
  n = 55;
  cout << "n =<< n << endl;
}
```

سطر اول، iostream باید به iostream.h تغییر یابد.

سطر ;n = 55 به int n = 55 تغییر یابد.

سطر ;cout << "n = << n << endl به صورت ;cout << "n = " << n << endl تغییر یابد. در انتها، قبل از } دستور ;return 0 قرار گیرد، چون نوع تابع اصلی (main)، int است.

مثال ۷ – ۲. کد زیر چه اشتباهی دارد؟

```
int  A = 12;
a++;
```

خط اول A را تعریف کرده، مقدار 12 را به آن تخصیص داده است و خط دوم به a یک واحد اضافه می‌نماید (a، تعریف نشده است). چون شناسه‌های A و a در ++C با هم فرق دارند (++C به حروف بزرگ و کوچک حساس است).

مثال ۱ – ۳. برنامه‌ای که عددی را خوانده، تشخیص می‌دهد مثبت است یا منفی.

متغیر	هدف
num	عدد خوانده شده

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int num;
  cout << "Please enter a number:";
  cin >> num;
  if ( num >= 0  )
  {
    cout << num << " is positive";
  }
  else
  {
    cout << num  << "is negative";
  }
  getch();
}
```

مثال ۲ - ۳. برنامه‌ای که دو عدد را خوانده، تعیین می‌کند باقی‌مانده تقسیم عدد اول بر عدد دوم زوج است یا خیر؟

متغیر	هدف
a	عدد اول
b	عدد دوم

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int a, b;
  cout << "Please enter two numbers:";
  cin >> a >> b;
  if ( ( a % b ) % 2 == 0 )
  {
```

```
    cout << endl<< a << "Mod" << b << "=" << a % b << "is" << "Even";
  }
  else
  {
   cout << endl<< a << "Mod" << b << "=" << a % b << "is"  << "Odd";
  }
  getch();
}
```

مثال ۵ – ۳. برنامه‌ای که سه ضلع مثلثی را خوانده، تشخیص می‌دهد آیا سه ضلع اولاً تشکیل مثلث را می‌دهند. اگر تشکیل مثلث را می‌دهند، آیا مثلث متساوی‌الساقین است یا خیر؟

متغیر	هدف
a	ضلع اول
b	ضلع دوم
c	ضلع سوم

```
#include "iostream.h"
#include "conio.h"
void main()
{
 int a, b, c;
 cout << "Enter a, b, c:";
 cin >> a >> b >> c;
 if ( a + b > c && b + c > a && a + c > b)
 {
    if (a == b || b == c || c == a)
    {
     cout << "Equal teral";
    }
    else
    {
      cout << "Not Equal teral";
    }
 }
```

```
 else
 {
  cout << "Not teral";
 }
 getch();
}
```

برنامه ای که عددی را خوانده ، بزرگ ترین رقم آن را نمایش دهد ؟

توضیح: در این برنامه، ابتدا بزرگ‌ترین رقم ۱- در نظر گرفته شده است (چون همه ارقام بین 0 تا ۹ هستند، پس از ۱- بزرگ‌تر هستند). سپس، عددی را خوانده تا زمانی که عدد بزرگ‌تر از صفر است (شرط while)، ارقام عدد را یکی یکی جدا می‌نماید (r = n % 10). این رقم را با رقم max مقایسه می‌کند، اگر رقم جدا شده بزرگ‌تر از max باشد، آن را در max قرار می‌دهد. سپس، عدد را تقسیم بر ۱۰ کرده تا رقم یکان آن حذف شود و این روند را ادامه می‌دهد. در پایان max را نمایش می‌دهد.

متغیر	هدف
n	عدد خوانده شده
r	رقم یکان عدد n در حلقه
max	بزرگ‌ترین رقم

```
#include "iostream.h"
#include "conio.h"
void main()
{
 int max = -1;
 long n;
 cout << "Enter n:";
 cin >> n;
 while (n > 0)
 {
  int r= n % 10;
  if (r > max) max = r;
  n /= 10;
 }
 cout << "Max  is " << max;
 getch();
}
```

مثال ۸ – ۳. برنامه‌ای که اعداد ۱ تا ۵۰ و مجموع آن‌ها را نمایش می‌دهد.

متغیر	هدف
sum	مجموع مقادیر ۱ تا ۵۰
i	شمارنده

```
#include "iostream.h"
#include "conio.h"
void main()
{
   int sum = 0;
   for( int i= 1; i <= 50; i++)
   {
     cout <<  "\t" << i;
     sum += i;
   }
   cout << endl << "Sum is " << sum;
   getch();
}
```

مثال ۶ – ۳. برنامه‌ای که نمره دانشجویی را از ورودی خوانده و با توجه با جدول زیر یکی از پیغام‌ها را نمایش می‌دهد (هدف این برنامه آشنایی با ساختار if تودرتو است). در این برنامه grade نمره دانشجو می‌باشد.

پیغام	نمره
Fail	0 – 70
Good	71 – 80
Very Good	81 – 90
Excellent	90 – 100
Invalid number	<0 , >100

```
#include "iostream.h"
#include "conio.h"
void main()
{
  float grade;
  cout << "Please enter a grade:";
  cin >> grade;
  if ( grade >= 0 && grade <= 70 )
  {
    cout << "Fail";
  }
  else if ( grade > 70 && grade <= 80 )
  {
    cout << "Good";
  }
  else if ( grade > 81 && grade <= 90 )
  {
    cout << "Very Good";
  }
    else if ( grade > 91 && grade <= 100 )
  {
    cout << "Excellent";
  }
  else
  {
    cout << "Invalid number";
  }
  getch();
}
```

مثال۱. برنامه‌ای که عددی را خوانده، اگر عدد ۱ بود، شنبه، وگرنه اگر عدد ۲ بود، یک شنبه، وگرنه اگر عدد ۳ بود، دوشنبه و این روند ادامه می‌یابد و اگر عدد ۷ بود، جمعه را نمایش می‌دهد. در غیر این صورت، invalid number را نمایش می‌دهد.

متغیر	هدف
n	عدد خوانده شده

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int n;
  cout << "Please enter a number:";
  cin >> n;
  switch(n)
```

```
  {
    case 1:
      cout << "Saturday";
      break;
    case 2:
      cout << "Sunday";
      break;
    case 3:
      cout << "Monday";
      break;
    case 4:
      cout << "Tuesday";
      break;
    case 5:
      cout << "Wednesday";
      break;
    case 6:
      cout << "Thursday";
      break;
    case 7:
      cout << "Friday";
      break;
    default:
      cout << "Invalid number";
      getch();
      return;
 }
 getch();
}
```

مثال ۲ . برنامه‌ای که سن فردی را خوانده، اگر سن کوچک‌تر از ۱۸ باشد، عبارت "you are a child" را نمایش می‌دهد، اگر سن بین ۱۸ تا ۶۰ باشد، عبارت "you are an adult" را نمایش می‌دهد، اگر سن بزرگ‌تر یا مساوی ۶۰ باشد، عبارت "you are a cenior citizen" را چاپ می‌کند، اگر سن کوچک‌تر از صفر وارد شود، پیغام error را نمایش می‌دهد.

متغیر	هدف
age	سن

```
#include "iostream.h"
#include "conio.h"
void main()
{
  unsigned int age;
  cout << "Please enter age:";
  cin >> age;
  if (age < 0 )
  {
    cout << " Error";
  }
  else if (age < 18 )
  {
    cout << "You are a child";
  }
  else if ( age >= 18 && age < 60)
  {
    cout << "You are an adult";
  }
 else
 {
    cout << "You are a cenior citizen";
}
 getch();
}
```

}

مثال ۱۳. برنامه‌ای که تمام اعداد دو رقمی را نمایش دهد که بر مجموع ارقام خودشان بخش پذیراند.

```
#include "iostream.h"
#include "conio.h"
void main()
{
int a, b, i;
for (i=10; i <= 99; i++)
{
  a = i / 10;
  b = i % 10;
  if(i% (a+b) == 0) cout << i << " ";
}
getch();
}
```

مثال ۱۸. اشتباه دستورات زیر را بی‌یابید:

```
if  x > y     max = x
else     max = y;
```

این دستورات به صورت زیر تغییر می‌یابند:

```
if  (x > y)    max = x;
else     max = y;
```

مثال ۱۹. اشتباه دستورات زیر چیست؟

```
int num;
cout << "enter a number:"
cin >> num;
if (num < 0)
     cout << "number is negative. try again.\n";
     cin >> num;
else
     cout << "number is " << num;
```

چون دستورات بخش if بیش از یکی است، پس باید بین } و { قرار گیرند.

مثال ۲۰. چه اشتباهی در دستورات زیر وجود دارد؟

```
if (x = 10) cout << "x = 22 \n";
  else  cout << "x != 22 \n";
```

چون این دستور می‌خواهد برابری را چک کند، بنابراین باید دستور if به صورت if (x == 10) بیان شود.

مثال ۲۴. برنامه‌ای که تعدادی عدد صحیح که آخرین آن‌ها صفر است را از ورودی دریافت کرده، تعداد اعداد زوج و فرد را نمایش می‌دهد.

متغیر	هدف
n	اعداد ورودی
e	تعداد اعداد زوج
o	تعداد اعداد فرد

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int n, e = 0,o = 0;
 cout <<"Enter n:";
 cin>>n;
 while(n > 0)
  {
   if(n%2==0) e++;
   else o++;
   cout << "Enter another n:";
   cin >> n;
  }
 cout << "\tNumber of  even= " << e;
 cout << "\tNumber of  odd= " << o;
 getch();
 return;
}
```

مثال ۲۶. برنامه‌ای که عدد صحیح و مثبت n را خوانده، میانگین ارقام آن را نمایش می‌دهد.

متغیر	هدف
n	عدد خوانده
sum	مجموع ارقام
k	تعداد ارقام
avg	میانگین ارقام

```
#include<iostream.h>
#include<conio.h>
void  main()
{
 long n;
 int sum = 0,k = 0;
 float ave = 0;
 cout << "Enter  n:";
 cin >> n;
 while(n > 0)
 {
  d = n % 10;
  sum += d;
  n = n / 10;
  k++;
 }
  if(k!=0) ave=sum/k;
  cout << "Ave = " << ave;
  getch();
  return;
}
```

مثال ۱۱ – ۲. نتیجه اجرای دستورات زیر چیست؟

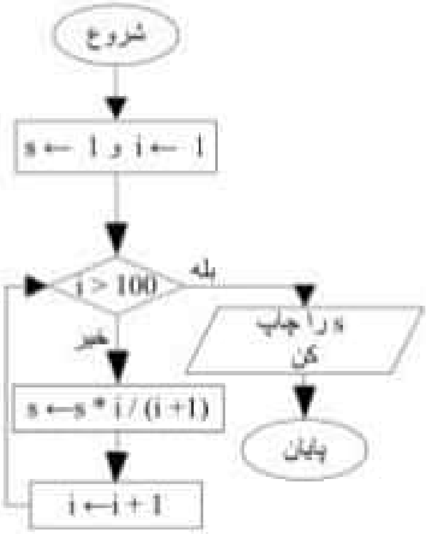
```
int x = 5;
int y = x++ * 5;
```

دستور اول، x را از نوع int تعریف کرده، ۵ را در آن قرار می‌دهد. دستور دوم، ابتدا مقدار x (۵) را در ۵ ضرب کرده، در y قرار می‌دهد (یعنی، ۲۵ را در y قرار می‌دهد) و سپس، به x یک واحد اضافه می‌نماید (یعنی، x برابر ۶ خواهد شد).

مثال ۲۹ - ۱. الگوریتم و فلوچارتی که حاصل ضرب ۱۰۰ جمله عبارت زیر را نمایش می‌دهد:

$$s=\frac{1}{2}\times\frac{3}{4}\times\frac{5}{4}\times\dots$$

۱. شروع
۲. یک را در s قرار بده
۳. یک را در i قرار بده (i: شمارنده جملات)
۴. اگر i > 100 آنگاه برو به ۸
۵. $s\times {}^{i}\!/\!_{(i+1)}$ را در s قرار بده
۶. i +1 را در i قرار بده
۷. برو به ۴
۸. s را چاپ کن
۹. پایان



این الگوریتم یک را در s (حاصل ضرب جملات سری) قرار می‌دهد و سپس یک را در i (شمارنده جملات سری) قرار می‌دهد و تا زمانی که i بزرگ‌تر از ۱۰۰ شود، i تقسیم بر i + 1 کرده تا جملات $\frac{1}{2}$، $\frac{2}{3}$ و ... را تولید کند. سپس آن‌ها را در s ضرب کرده، در s قرار می‌دهد. به محض این که i بزرگ‌تر از ۱۰۰ شود، s را نمایش می‌دهد.

مثال ۹ – ۲. نتیجه اجرای دستورات زیر چیست؟

```
int x = 5, y = 3;
(x ++, y += x, x += y)
```

دستور اول، متغیرهای x و y را از نوع int تعریف می‌کند. در دستور دوم به ترتیب عبارات ++ x، y += x و x += y انجام خواهند شد. یعنی، ابتدا، به مقدار x یک واحد اضافه می‌شود (x برابر ۶ خواهد شد). سپس، عبارت y += x انجام می‌گردد. یعنی، y برابر با ۳ + ۶ (۹) خواهد شد و در پایان، عبارت x += y انجام می‌گردد. یعنی، x برابر با ۹ + ۶ (۱۵) خواهد شد.

مثال ۳. اشتباهات دستورات زیر چیست؟

```
A: cin << n;
B: if  a < b min = a
   else  min = b
c: if (a < 0)
     cout << " that is negative";
     cin >> n
  else
    cout << " n is" <<  n;
d: if (x = 0) cout << x<< "= 0";
    else  cout << x < "!=0";
e: if (x < y < z)    cout << x << "<"  << y << "<" << z;
```

A: شیء cin به صورت مقابل به کار می‌رود: cin >> متغیر;

یعنی، به جای عملگر >> باید از عملگر << استفاده شود. پس، داریم: cin >> n;

B: شرط if باید در داخل پرانتز قرار گیرد و بعد از دستور بخش if ; قرار می گیرد. پس داریم:

```
if (a < b)       min = a;
else             min = b;
```

C: چنانچه بخواهید چند دستور را در بخش if یا else قرار دهید، آن‌ها را در بلاک { و } قرار می‌دهید:

```
if (a < 0)
  {
   cout << " that is negative";
   cin >> n;
  }
else     cout << " n  is" <<  n;
```

D: علامت برابر == است. پس، داریم:

```
if (x == 0) cout << x << " = 0 ";
 else  cout << x << "! = 0";
```

E: عبارت داخل پرانتز if باید به صورت زیر تغییر یابد:

```
if (x < y && y < z)   cout << x <<  " < "  << y <<  " < " << z;
```